Vizuális, eseményvezérelt programozás I.

Visual Studio ismétlés

Grafikus felület tervezése

Vezérlők alapvető tulajdonságai, metódusai, eseményei

Hibakezelés

Hallgatói tájékoztató

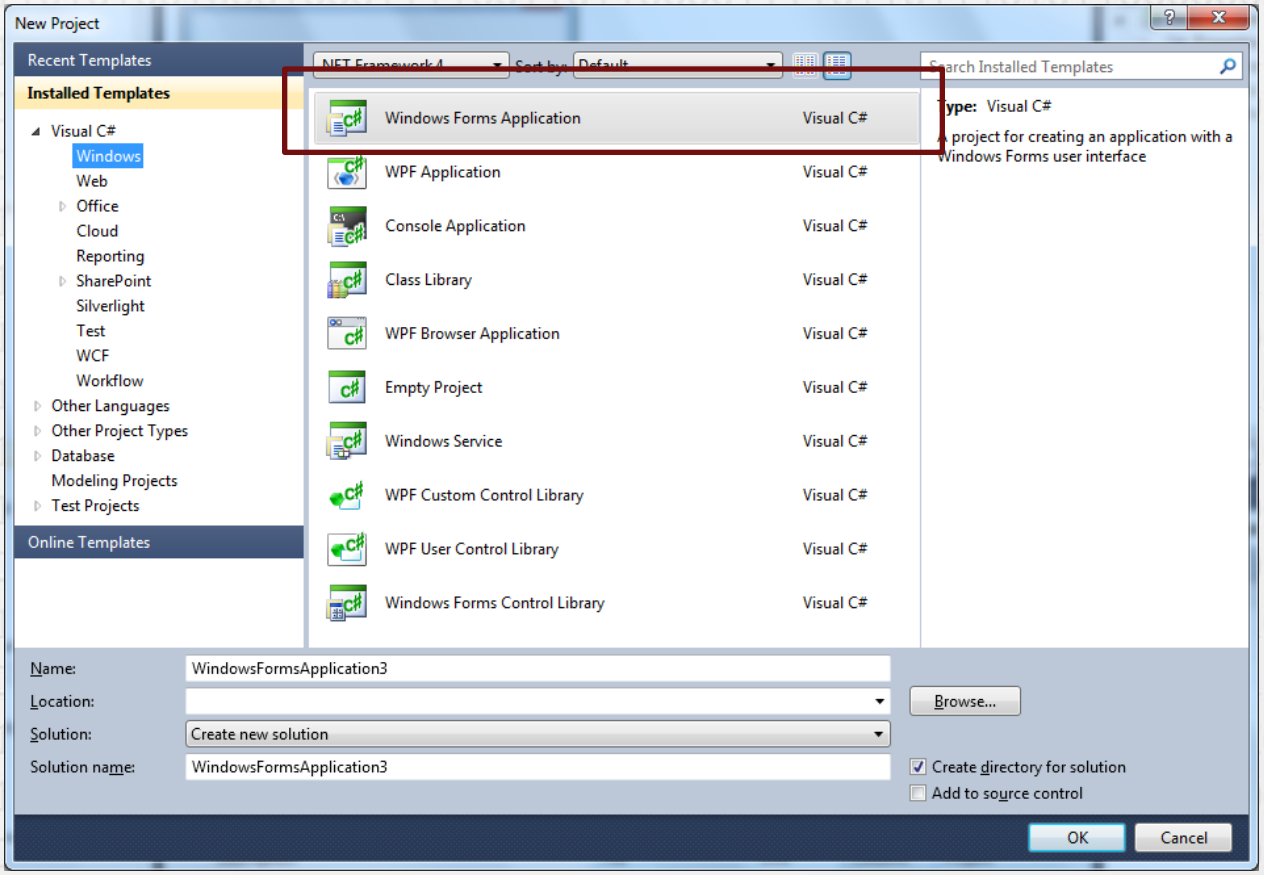
A jelen bemutatóban található adatok, tudnivalók és információk a számonkérendő anyag vázlatát képezik. Ismeretük szükséges, de nem elégséges feltétele a sikeres zárthelyinek, illetve vizsgának.

Sikeres zárthelyihez, illetve vizsgához a jelen bemutató tartalmán felül a kötelező irodalomként megjelölt anyag, a gyakorlatokon szóban, illetve a táblán átadott tudnivalók ismerete, valamint a gyakorlatokon megoldott példák és az otthoni feldolgozás céljából kiadott feladatok önálló megoldásának képessége is szükséges.

Projektek és megoldások

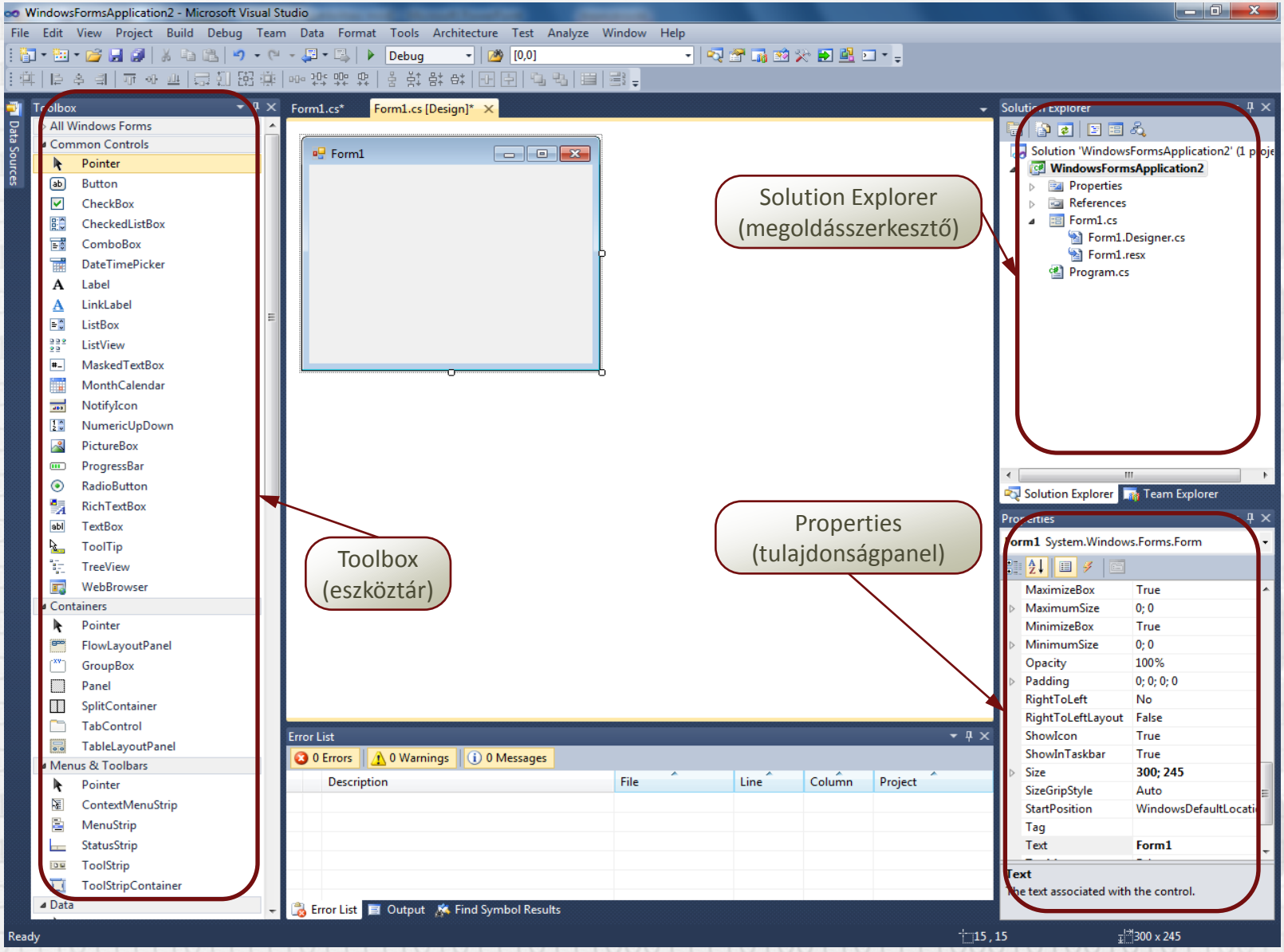
- **Projekt („project”)**
  - A projekt egy futtatható programhoz vagy más típusú szoftvermodulhoz tartozó, együtt kezelt szoftverelemek (többségében fájlok) összessége
    - C# forráskódok („source code”) [*.cs]
    - Hivatkozások („references”)
    - Beállítások („settings”) [*.settings]
    - Konfigurációs fájlok („configuration”) [*.config]
    - Egyéb erőforrások („resources”) [*.resx, *.rc, *.resources]
  - A projekthez tartozó elemek mappák létrehozásával hierarchikus fastruktúrába rendezhetők.
  - C# projekt kiterjesztése: *.csproj
- **Megoldás („Solution”)**
  - Több összefüggő projekt együttes kezelését teszi lehetővé
  - A projektek virtuális mappák segítségével hierarchikus fastruktúrába is rendezhetők.
  - Megoldások kiterjesztése: *.sln

Új projekt indítása



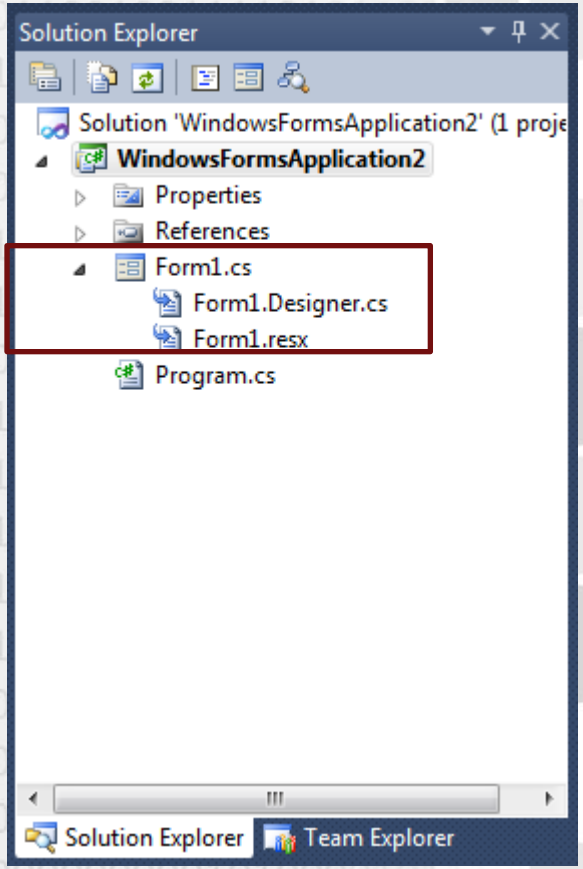
- **Console Application** – konzolos alkalmazás
- **Windows Forms Application** – grafikus Windows alkalmazás
- **Windows Service** – háttérben futó Windows rendszerszolgáltatás
- **Class Library** – osztálykönyvtár
- **Windows Forms Control Library** – Windows vezérlők gyűjteménye
- **Empty Project** – Ehhez a projekttípushoz kézzel kell a megfelelő elemeket hozzáadni

Grafikus felhasználói felület tervezése



Az ablak fájljai

- **Form1.cs**
  - Ebbe a fájlba kerül a felhasználói kód, kézzel általában csak ezt módosítjuk
- **Form1.Designer.cs**
  - Automatikusan generált kód, amely a Designerben (az ablakszerkesztőben) összeállított ablak kinézetét generálja futásidőben
- **Form1.resx**
  - Az ablakkal kapcsolatos erőforrásokat tároló XML formátumú fájl
    - pl. háttérképek vagy ikonok, hosszú stringek, lokalizációs stringek kerülnek bele

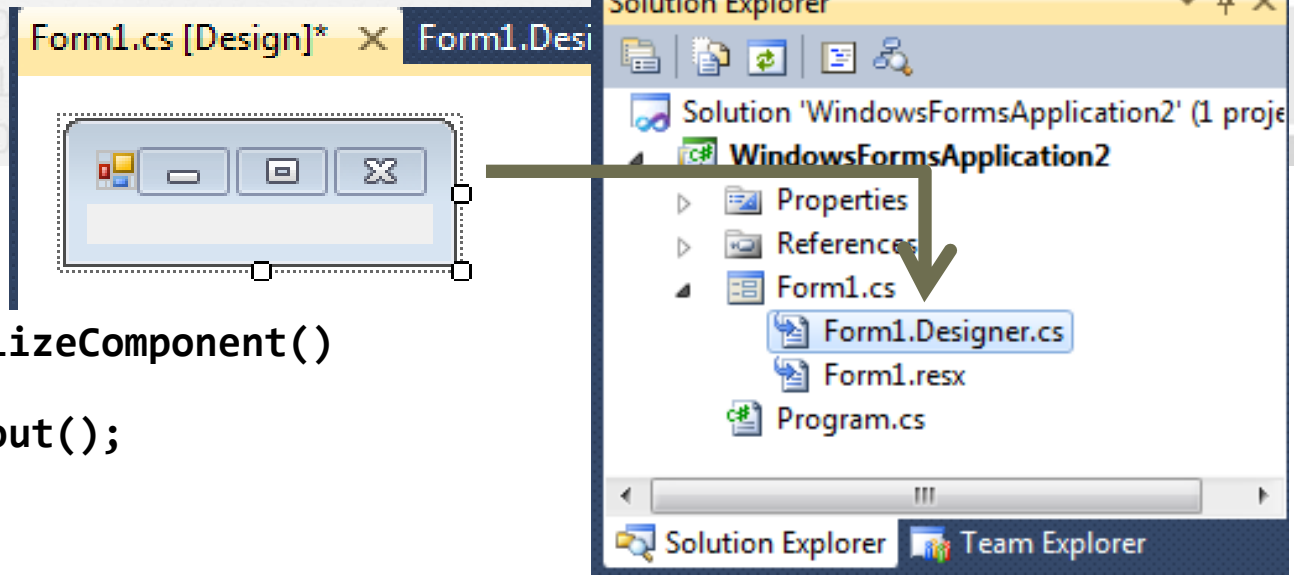


Felbontott típusok

- **A Form1 osztály felbontott osztály (partial class)**
  - Partial class:
    - Minden részt a partial kulcsszóval kell megjelölni
    - Előnye, hogy a típusok úgy oszthatók meg több programozó vagy automatikus kódgenerátor között, hogy fizikailag nem kell osztozniuk a forrásfájlokon
      - Különválasztható (és ezáltal külön fejleszthető és verzionálható) az osztályok automatikusan, illetve kézzel előállított része
      - Különválasztható az egyes osztályok kódján dolgozó fejlesztőcsapatok munkája
  - Osztályok, struktúrák, interface-ek lehetnek ilyenek (+partial void)
- **A felbontott típusok elemeit a C# fordító összefésüli**
  - A fordító úgy kezeli az elemeket, mintha egy fájlban, egy típusdefinícióként hoztuk volna létre őket

Designer.cs

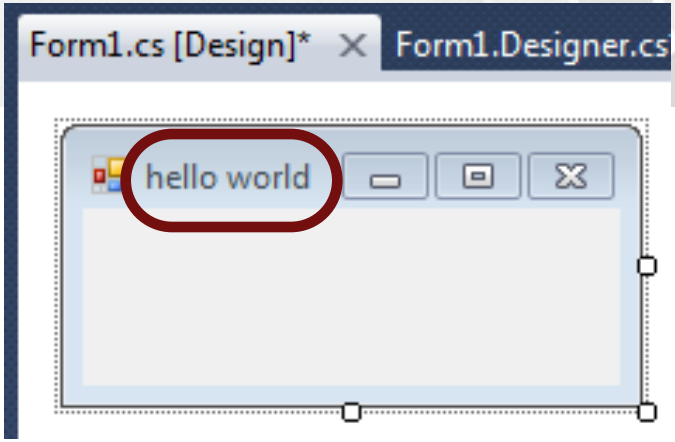
- **Az ablak esetében a Form1.Designer.cs-ben lévő automatikusan generált kód és a Form1.cs-ben lévő általunk megírt kód együtt alkotják a Form1 osztályt**
- **A Form1.Designer.cs az ablakszerkesztőben való megnyitáskor és ottani változtatások hatására mindig újragenerálódik**




```
partial class Form1
{
    ...
    private void InitializeComponent()
    {
        this.SuspendLayout();
        //
        // Form1
        //
        this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
        this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
        this.ClientSize = new System.Drawing.Size(116, 14);
        this.Name = "Form1";
        this.Load += new System.EventHandler(this.Form1_Load);
```

Designer.cs

- Az ablak tulajdonságai az InitializeComponent() függvényben állnak be a kívánt értékre
- Az InitializeComponent() az ablak konstruktorából hívódik

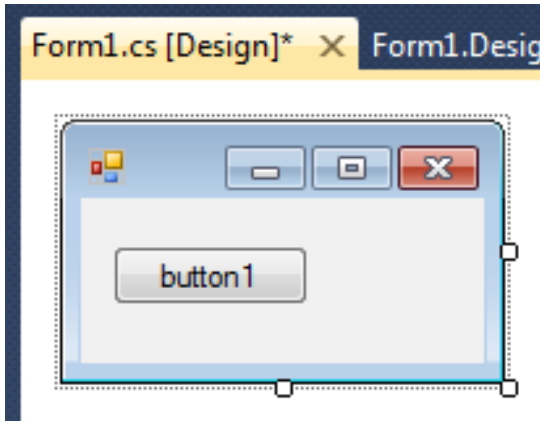



```
partial class Form1
{
    private void InitializeComponent()
    {
        this.SuspendLayout();
        //
        // Form1
        //
        this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
        this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
        this.ClientSize = new System.Drawing.Size(191, 63);
        this.Name = "Form1";
        this.Text = "hello world";
        this.Load += new System.EventHandler(this.Form1_Load);
        this.ResumeLayout(false);
    }
}
```

Designer.cs

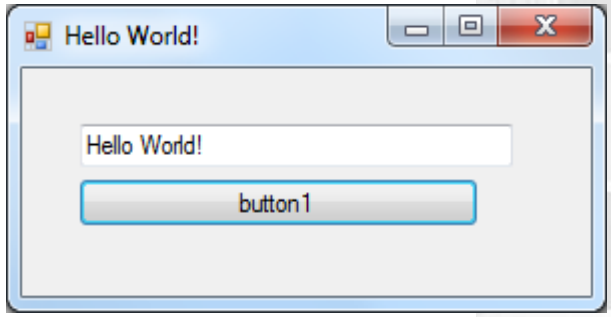
- **Az ablakra helyezett komponensek a Form1.Designer.cs-ben megjelennek mint az ablak tagváltozói**
  - Kivéve, ha GenerateMember == false (később)
- **Tulajdonságaik szintén az InitializeComponent()-ben állnak be**

```
partial class Form1
{
    ...
    private void InitializeComponent()
    {
        ...
        this.button1 = new System.Windows.Forms.Button();
        this.SuspendLayout();
        //
        // button1
        //
        this.button1.Location = new System.Drawing.Point(12, 18);
        this.button1.Name = "button1";
        this.button1.Size = new System.Drawing.Size(75, 23);
        this.button1.TabIndex = 0;
        this.button1.Text = "button1";
        this.button1.UseVisualStyleBackColor = true;
        ...
    }
    ...
    private System.Windows.Forms.Button button1;
}
```

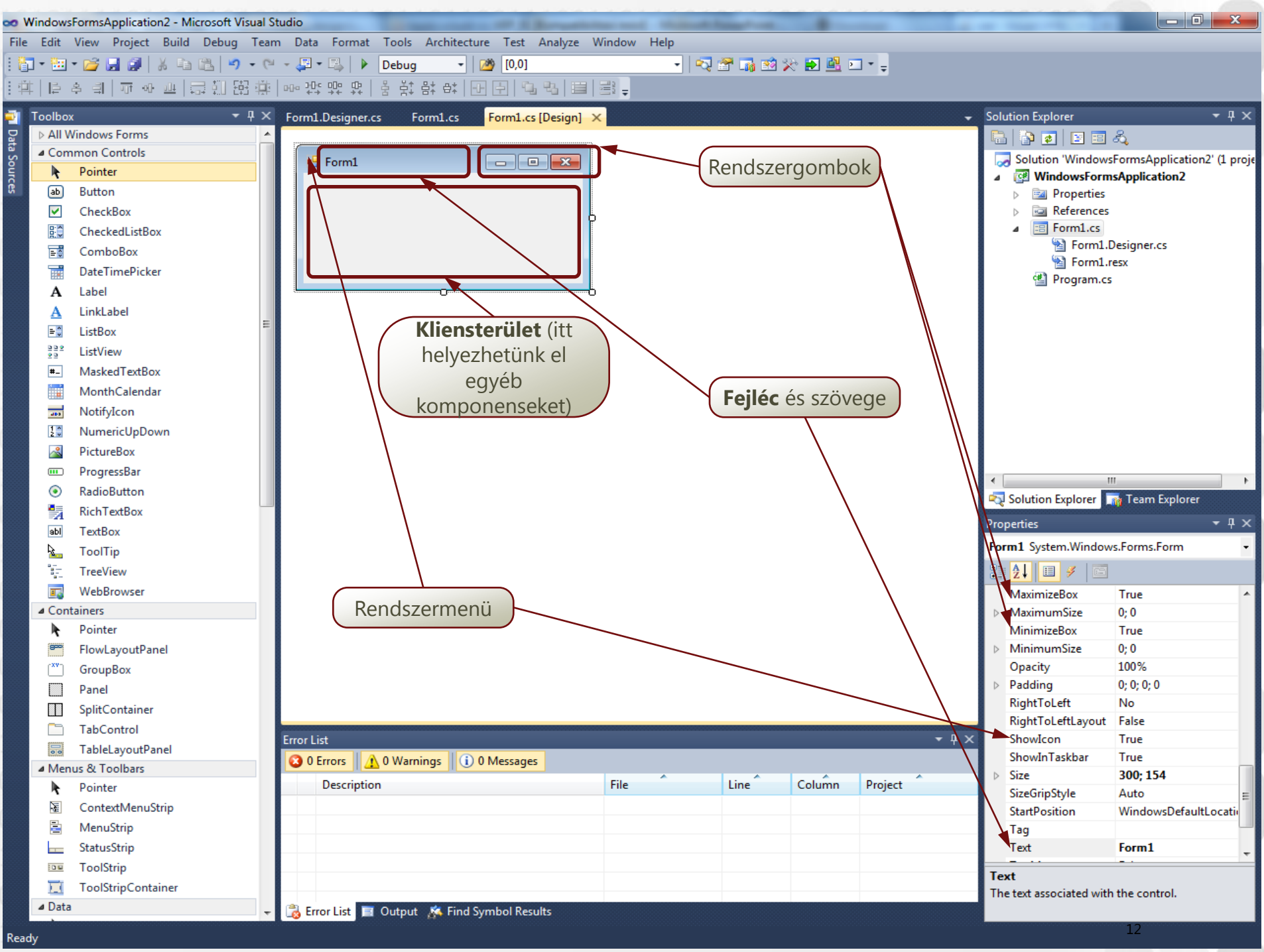


Egyszerű példaalkalmazás készítése

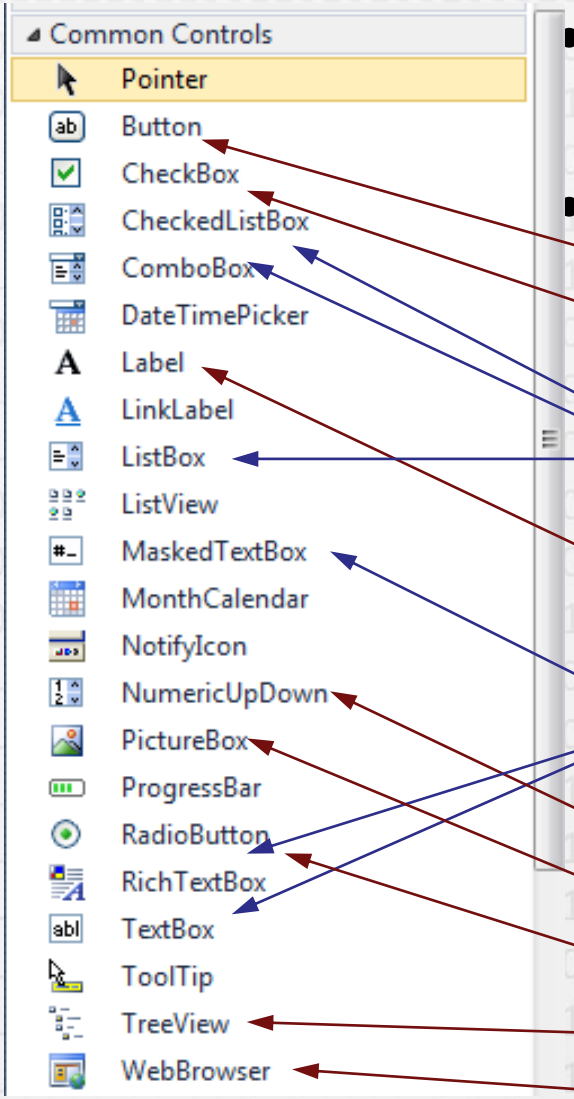
```
namespace WindowsFormsApplication2
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            Text = "Hello World!";
            button1.Width = 200;
            textBox1.Text = "Hello World!";
        }
    }
}
```



- **Bármilyen kódnak, amiben hozzányúlunk a komponensekhez, az InitializeComponent() után kell lefutnia**
  - Különben az InitializeComponent()-beli beállítások felülírják a mieinket
- **A fenti megoldás OOP alapelveknek megfelel, de a vizuális alkalmazásfejlesztés követelményeinek nem. Hasonló célokra a konstruktor helyett többnyire eseménykezelőket használunk (később)**


WindowsFormsApplication2 - Microsoft Visual Studio
File Edit View Project Build Debug Team Data Format Tools Architecture Test Analyze Window Help
Debug
[0,0]
Toolbox
All Windows Forms
Common Controls
Pointer
Button
CheckBox
CheckedListBox
ComboBox
DateTimePicker
Label
LinkLabel
ListBox
ListView
MaskedTextBox
MonthCalendar
NotifyIcon
NumericUpDown
PictureBox
ProgressBar
RadioButton
RichTextBox
TextBox
ToolTip
TreeView
WebBrowser
Containers
Pointer
FlowLayoutPanel
GroupBox
Panel
SplitContainer
TabControl
TableLayoutPanel
Menus & Toolbars
Pointer
ContextMenuStrip
MenuStrip
StatusStrip
ToolStrip
ToolStripContainer
Data
Data Sources
Form1.Designer.cs
Form1.cs
Form1.cs [Design]
Form1
Rendszergombok
Kliensterület (itt helyezhetünk el egyéb komponenseket)
Fejléc és szövege
Rendszermenü
Solution Explorer
Solution 'WindowsFormsApplication2' (1 proje
WindowsFormsApplication2
Properties
References
Form1.cs
Form1.Designer.cs
Form1.resx
Program.cs
Solution Explorer
Team Explorer
Properties
Form1 System.Windows.Forms.Form
MaximizeBox True
MaximumSize 0; 0
MinimizeBox True
MinimumSize 0; 0
Opacity 100%
Padding 0; 0; 0; 0
RightToLeft No
RightToLeftLayout False
ShowIcon True
ShowInTaskbar True
Size 300; 154
SizeGripStyle Auto
StartPosition WindowsDefaultLocati
Tag
Text Form1
Text
The text associated with the control.
Error List
0 Errors
0 Warnings
0 Messages
Description
File
Line
Column
Project
Error List
Output
Find Symbol Results
Ready

Beépített komponensek



- **Pointer: nem komponens, ezzel váltunk vissza szerkesztő módba**
- **Közönséges vezérlők (Common Controls)***

Nyomógomb

Jelölőnégyzet

Listák:
ListBox: egyszerű lista
ComboBox: legördülő lista
CheckedListBox: egyszerű lista kipipálható elemekkel

Címke (szöveg az ablakon)

Szövegdobozok:
TextBox: sima szövegbeviteli mező
MaskedTextBox: adott formátumú bevitelt biztosít
RichTextBox: formázható (szín, betűtípus stb.) szövegbevitelt biztosít

Numerikus beviteli mező

Kép

Rádiógomb

Fanézet

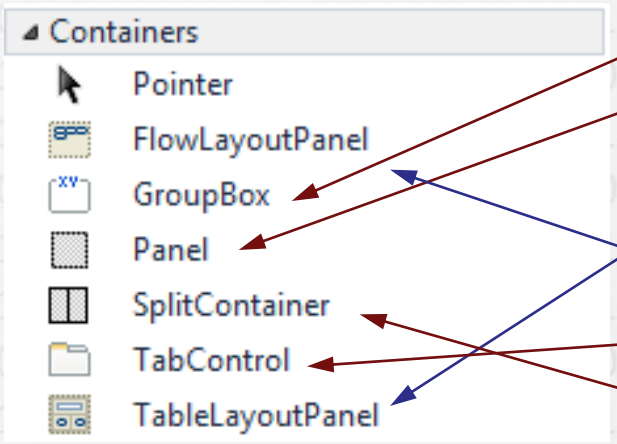
Webböngésző**

* A beépített komponensek tárgyalása nem teljes körű, a legfontosabb vezérlők bemutatására szorítkozunk.

** A WebBrowser komponens gyakorlatilag a telepített Internet Explorer egy példánya, ennek összes előnyével és hátrányával.

Beépített komponensek

- **Csoportosító vezérlők (Containers)**
  - Az ablak szerkezetének kialakításában játszanak szerepet, más vezérlők csoportosítására szolgálnak



Csoport*

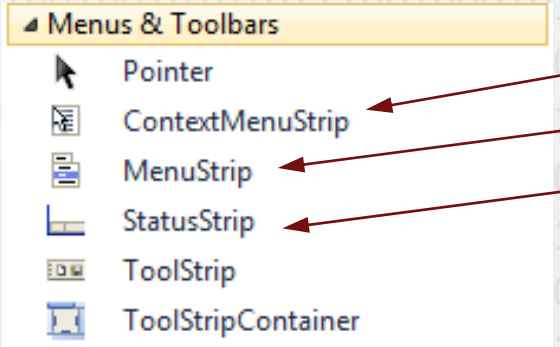
Panel*

Automatikus elrendezők:
FlowLayoutPanel: egymás után rendezi el a belehelyezett vezérlőket
TableLayoutPanel: táblázatszerű elrendezést biztosít

Füles vezérlő

Méretezhető kétfelé választó vezérlő

- **Menük és eszközkészletek (Menus & Toolbars)**



Kontextusmenü

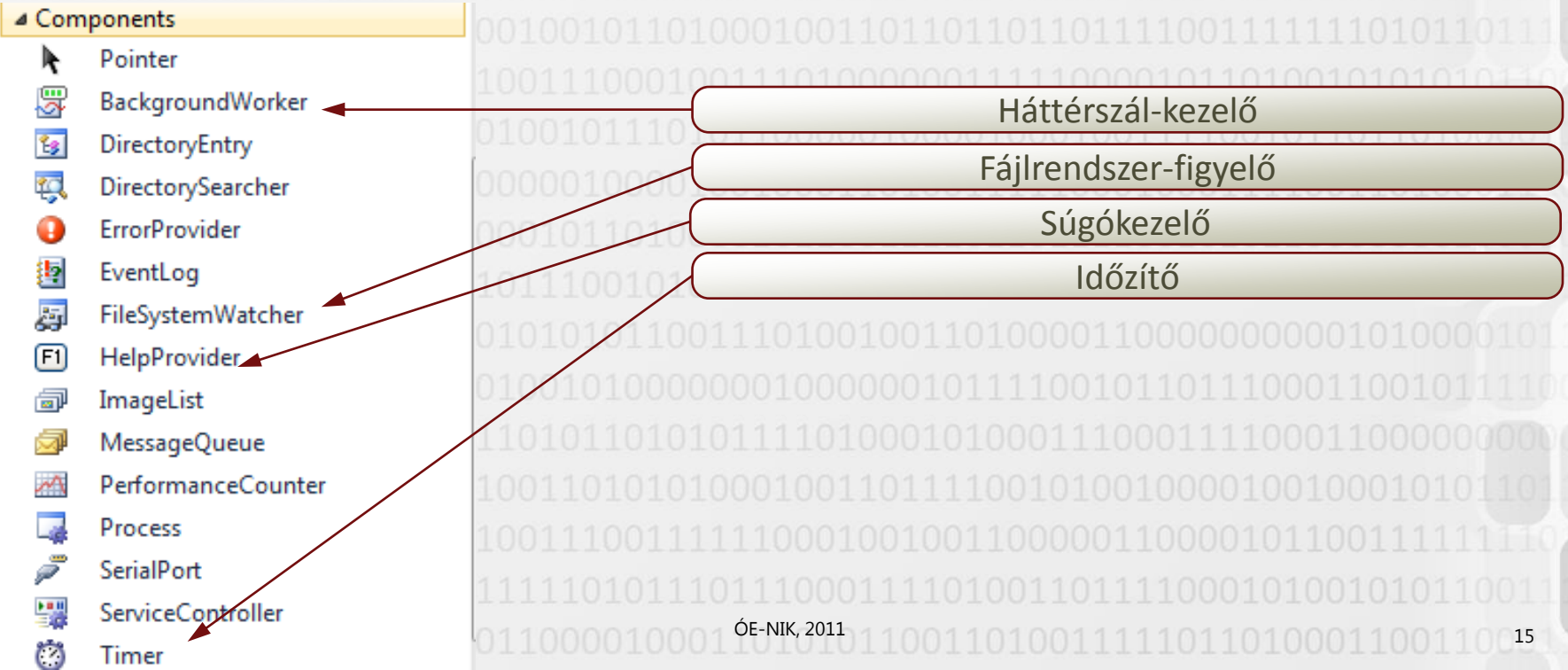
Menü

Státuszsor

* A csoport és a panel lényegében ugyanazt a funkciót valósítja meg (egyszerű grafikus csoportosítás). Fő különbségek: a csoportnak van címe, a panel pedig tud scrollozni, ha a benne elhelyezett tartalom nagyobb, mint a panel mérete.

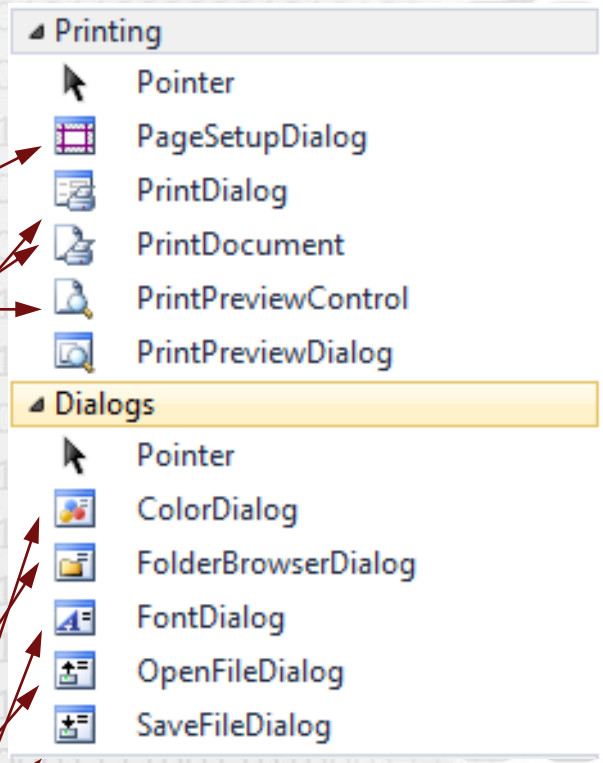
Beépített komponensek

- **Adatkezelő komponensek (Data)**
  - Adatrács és más komponensek (főként adatbázis-kezelésben használatosak)
- **Komponensek (Components)**
  - Az ablak nem vizuális elemei
  - Időzítő, fájlrendszer-figyelő, háttérszál-kezelő, súgókezelő stb.

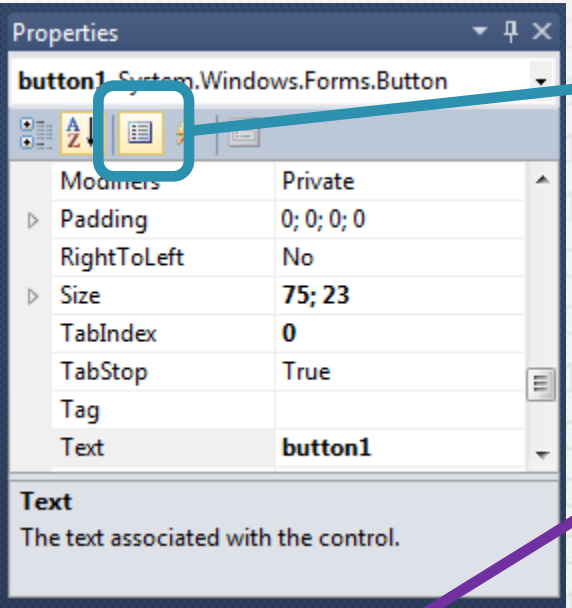


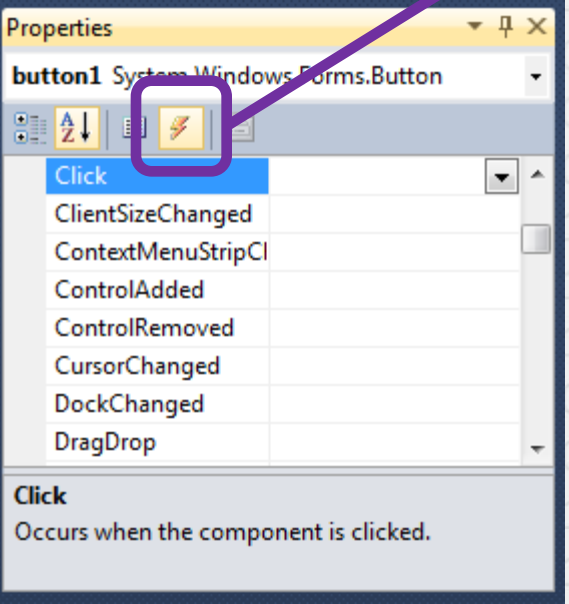
Beépített komponensek

- **Nyomtatással kapcsolatos komponensek (Printing)**
  - Oldalbeállító ablak
  - Nyomtatási kép
  - Nyomtatható dokumentum
  - Nyomtatási párbeszédablak
- **Párbeszédablakok (Dialogs)**
  - Szintén nem jelennek meg az ablakon, de procedurálisan megjeleníthetők
    - Mappatallózó ablak
    - Fájl betöltése... ablak
    - Fájl mentése... ablak
    - Színválasztó ablak
    - Betűtípus ablak



Tulajdonságok és események





- **A szerkesztőben kijelölt komponens tulajdonságai a Properties ablakban állíthatók**
  - A módosított tulajdonságokat félkövér betű jelöli
  - A „Reset” helyi menüpont alaphelyzetbe állítja az adott tulajdonságot
- **Ugyanitt a komponens egyes eseményeihez ún. eseménykezelőket rendelhetünk**
  - Kettős kattintással új eseménykezelő hozható létre
  - Eseménykezelő törlése: „Reset” helyi menüponttal, vagy az eseménykezelő nevének kitörlése+enter
- **Eseménykezelőt hozzárendelni a komponensen való kettős kattintással is lehet, ez az *alapértelmezett eseményhez* rendel eseménykezelőt**

Eseménykezelők

- **Az eseménykezelő hozzárendelése szintén az ablak InitializeComponent() függvényébe kerül (bővebben PPT-n)**

```
this.button1.Click += new System.EventHandler(this.button1_Click);
```

- **Az eseménykezelő függvény felépítése**

```
private void button1_Click(object sender, EventArgs e)
{  }
```

  - Nincs visszatérési érték
  - sender paraméter:
    - Referencia az objektumra, amely az eseményt kiváltotta
    - Típusa: object, tehát bármely osztály egy példánya átadható benne (bővebben PPT-n!)
  - e paraméter:
    - Az eseménnyel kapcsolatos információkat tartalmazza (Pl. MouseMove eseménynél tartalmazza az egér aktuális koordinátáit, az éppen lenyomva tartott gombokat stb.)
    - Típusa: EventArgs és utódai (bővebben PPT-n!)

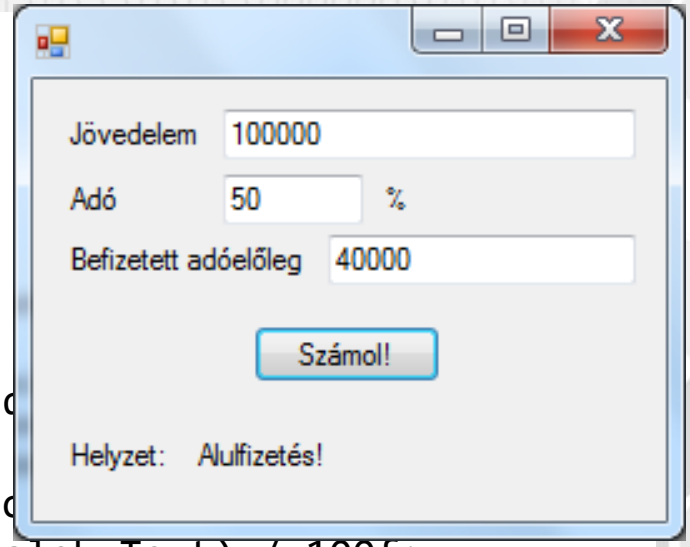
Egyszerű példaalkalmazás készítése II.

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }

    private void buttonSzamol_Click(object send
    {
        float jovedelem = float.Parse(textBoxJ
        float ado = float.Parse(textBoxAdoSzazalek.Text) / 100f;
        float adoeloleg = float.Parse(textBoxAdoEloleg.Text);

        float befizetendo = jovedelem * ado;

        if (befizetendo < adoeloleg)
            labelHelyzet.Text = "Túlfizetés!";
        else if (befizetendo > adoeloleg)
            labelHelyzet.Text = "Alulfizetés!";
        else
            labelHelyzet.Text = "Pontos befizetés!";
    }
}
```



Vezérlők alapvető tulajdonságai – azonosítás

- **Name**
  - A komponens egyedi megnevezése, amellyel hivatkozunk rá.
  - Erre a tulajdonságra az általános névadási szabályok vonatkoznak.
- **Parent**
  - A komponenst vizuálisan tartalmazó csoportosító vezérlő („szülő”).
  - Meghatározza a komponens koordinátarendszerének origóját.
- **Tag**
  - Tetszőleges objektum tárolható ebben a tulajdonságban (hivatkozás formájában).
    - A tulajdonság típusa object, tehát bármilyen osztály egy példányára hivatkozhat. (Bővebben PPT-n!)
  - Segítségével kiegészítő információk adhatók a komponenshez, külső vagy belső objektumok kapcsolhatók hozzá.

Vezérlők alapvető tulajdonságai – megjelenés

- **BackColor, ForeColor**
  - A komponens előtér- és háttérszínét határozzák meg.
  - Egyéni színek („Custom”), a weben gyakran használt színek („Web”), illetve rendszerszínkódok („System”) közül választhatunk.
- **BackgroundImage, BackgroundImageLayout**
  - A komponens háttérképe, amely a BackgroundImageLayout tulajdonság értéke alapján lehet a bal felső sarokba igazított („None”), ismétlődő („Tile”), középre igazított („Center”), méretre igazított („Stretch”) vagy egyéni méretezésű („Zoom”).
- **Cursor**
  - Az itt beállított egérmutató jelenik meg, amikor az egeret a komponens fölé húzzuk.

Vezérlők alapvető tulajdonságai – megjelenés

- **Font**
  - A komponens által megjelenített szöveg(ek) betűstílusát határozza meg.
  - Ha egyáltalán nem módosítjuk, akkor a komponens átveszi a szülője (vagy szülő hiányában tartalmazója) megfelelő tulajdonságának értékét.*
- **Image, ImageAlign**
  - A komponensen megjelenített kép, amelyet az ImageAlign tulajdonság segítségével vízszintesen, illetve függőlegesen is igazíthatunk.
- **Text, TextAlign**
  - A komponens szövege, amelyet a TextAlign tulajdonság segítségével vízszintesen, illetve függőlegesen is igazíthatunk.

* Az ilyen típusú tulajdonságokat a .NET keretrendszerben a „környezettől átvett tulajdonság” („ambient property”) kifejezés jelöli.

Vezérlők alapvető tulajdonságai – viselkedés

- **Enabled**
  - A komponens működtetésének engedélyezésére, illetve letiltására szolgál.
- **TabIndex, TabStop**
  - Ha a TabStop tulajdonság értéke igaz („true”), a komponens részt vesz a Tab billentyűvel végrehajtható bejárásban, méghozzá a TabIndex tulajdonság által megadott pozícióban.
- **Visible**
  - A komponens láthatóságának engedélyezésére, illetve letiltására szolgál.

Vezérlők alapvető tulajdonságai – elhelyezkedés

- **Anchor**
  - Segítségével „összeköthetjük” a komponenst a szülő bal/jobb/felső/alsó szegélyével.
  - Összekötés után a komponens és a szülő megfelelő széle mindig együtt mozog, tehát távolságuk átméretezéskor sem változik.
- **AutoSize, AutoSizeMode**
  - Ha az AutoSize tulajdonság értéke true, a komponens automatikusan a saját tartalma által megkívánt méretre nagyítja (AutoSizeMode == GrowOnly) vagy nagyítja és kicsinyíti (AutoSizeMode == GrowAndShrink) saját magát.
- **Dock**
  - Segítségével közvetlenül hozzákapcsolhatjuk a komponenst a szülő bal/jobb/felső/alsó szegélyéhez, vagy beállíthatjuk, hogy a komponens mindig teljesen töltse ki a szülő területét (Dock == Fill).

Vezérlők alapvető tulajdonságai (6) Elhelyezkedés

- **Location, Top, Left, Right, Bottom**
  - A komponens képpontokban számított pozícióját adja meg a szülőhöz viszonyított koordinátarendszerben.
  - A Left, Top tulajdonságokkal lekérdezhetjük vagy állíthatjuk a függőleges és vízszintes pozíciót.
  - A Right, Bottom tulajdonságokkal lekérdezhetjük a komponens aljának és jobb oldalának elhelyezedését (állítani nem lehet)
- **MaximumSize, MinimumSize**
  - A komponens képpontokban számított maximális, illetve minimális méreteit (szélességét és magasságát) adja meg.
- **Size**

  A komponens képpontokban számított méreteit (szélességét és magasságát) adja meg.

Vezérlők alapvető tulajdonságai – egyéb

- **GenerateMember**
  - Ha értéke true, a komponenshez a Visual Studio külön tagváltozót rendel a tartalmazó osztályban.
    - Ellenkező esetben a komponensre az InitializeComponent()-en kívüli kódból nem lehet majd hivatkozni.
- **Modifiers**
  - A komponens láthatósági szintje.
- **UseMnemonic**
  - Ha értéke true, a komponenshez gyorsbillentyűt rendelhetünk (a Text tulajdonságánál megadott szövegben az & karakter után álló betű).
    - Az & karakter ilyenkor nem jelenik meg a komponens szövegében.
- **UseWaitCursor**
  - Segítségével „várakozó kurzorra” állíthatjuk át a komponens egérmutatóját.

Vezérlők alapvető eseményei

- **Click**
  - A komponensre való kattintáskor hívódnak a hozzá rendelt eseménykezelők.*
- **Enter, Leave**
  - Akkor hívódnak a hozzá rendelt eseménykezelők, amikor a komponens megkapja (Enter), illetve elveszíti (Leave) a fókuszt.
- **KeyDown, KeyPress, KeyUp**
  - A komponens fókuszált** állapotában valamely billentyű lenyomásakor a KeyDown, felengedésekor a KeyUp, illetve a „normál”*** billentyűk megnyomásakor a KeyPress eseménykezelők hívódnak meg.
  - Sorrend: KeyDown, KeyPress, KeyUp

* A Visual Studio grafikus felületén nem tudunk egy eseményhez több eseménykezelőt rendelni (saját forráskódban viszont igen).

** Egy komponens akkor van fókuszált állapotban, ha az operációs rendszer hozzá irányítja a bemeneti eszközök eseményeit (egérmozgatás, billentyűlenyomás stb.) – lásd a Focus() metódus leírását.

*** „Normál” billentyűk: az ABC betűi, a számok és az írásjelek.

Vezérlők alapvető eseményei (2)

- **MouseDown, MouseMove, MouseUp**
  - Az egér valamelyik gombjának a komponens felett történő megnyomásakor a MouseDown, felengedésekor a MouseUp eseménykezelők hívódnak meg.
  - Az egérnek a komponens felett történő mozgatásakor a MouseMove eseménykezelők hívódnak meg. (Folyamatosan, sokszor a mozgás során.)
- **MouseEnter, MouseLeave**
  - Akkor hívódnak az eseménykezelői, amikor az egér a komponens fölé ér (MouseEnter), illetve elhagyja a komponens területét (MouseLeave).
- **Move, Resize**
  - Akkor hívódnak az eseménykezelői, ha a komponens pozíciója (Move) vagy mérete (Resize) megváltozott.

Vezérlők alapvető eseményei (3)

- **…Changed**
  - Az ezen eseményekhez tartozó eseménykezelők akkor hívódnak meg, ha a komponens megfelelő tulajdonságának értéke megváltozott.
    - Például az AutoSizeChanged eseménykezelői az AutoSize tulajdonság, a BackColorChanged eseménykezelői a BackColor tulajdonság megváltozásakor hívódnak meg, és így tovább.
  - Néhány gyakran használt …Changed eseménykezelő:
    - AutoSizeChanged
    - EnabledChanged
    - FontChanged
    - SizeChanged
    - VisibleChanged

Vezérlők alapvető metódusai (1)

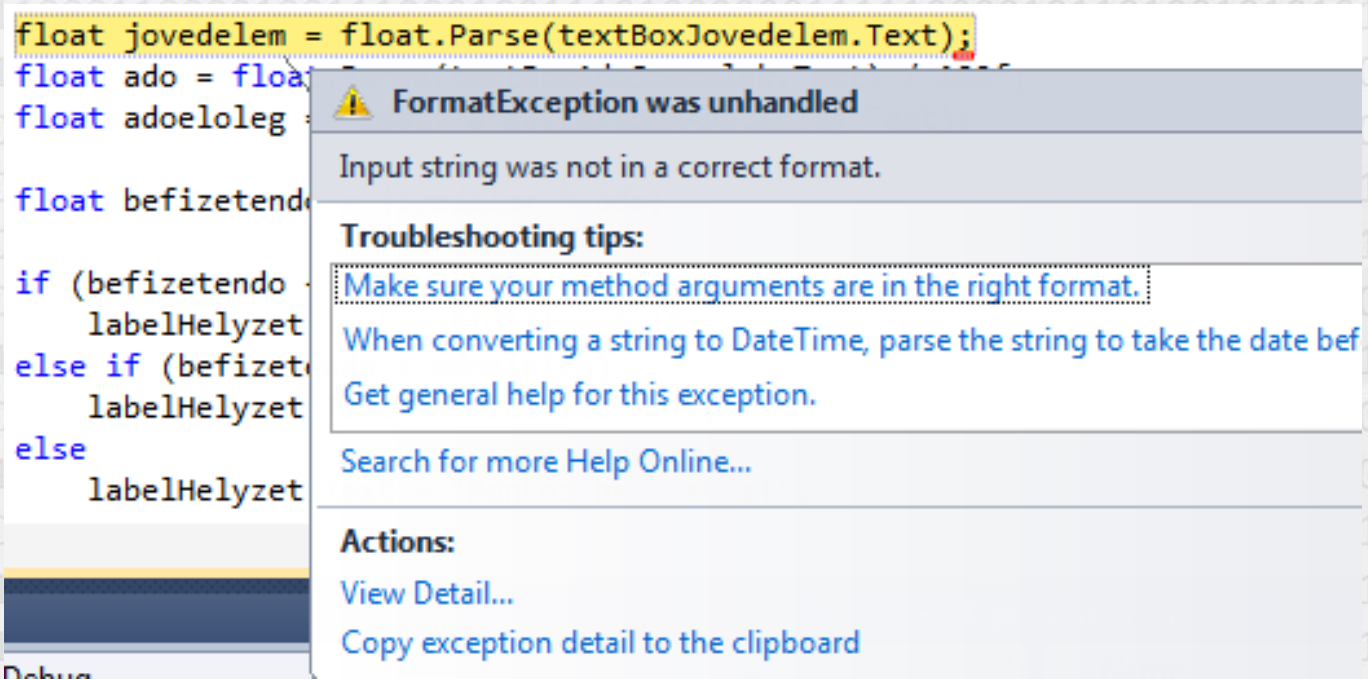
- **Hide(), Show()**
  - Komponens elrejtése, felfedése
- **Invalidate(), Update(), Refresh()**
  - Közvetve vagy közvetlenül a komponens újrarajzolását váltják ki:
    - Invalidate() – a komponens területét részben vagy teljesen újrarajzolandónak nyilvánítja, de rajzolást közvetlenül nem végez (az újrarajzolásra tehát a legközelebbi rajzolási ciklusban kerül sor)
    - Update() – a komponens korábban már újrarajzolandónak nyilvánított területeinek azonnali újrarajzolását váltja ki
    - Refresh() – a komponens teljes területét újrarajzolandónak nyilvánítja és azonnal el is végzi az újrarajzolást
- **BringToFront(), SendToBack()**
  - A komponenst az előtérbe (a többi komponens elé), illetve a háttérbe (a többi komponens mögé) helyezi.

Vezérlők alapvető metódusai (2)

- **Focus()**
  - A komponensre állítja a bemeneti fókuszt
- **Select()**
  - Kiválasztja a komponenst (nem minden komponens kiválasztható).
  - A tartalmazó komponens ActiveControl tulajdonságát is beállítja.
- **Scale()**
  - Adott tényezővel megszorozza a komponens (valamint az általa tartalmazott alkomponensek) minden méretét.
    - Kicsinyítésre és nagyításra is alkalmas.
- **SetBounds()**
  - A komponens helyét és méreteit állítja be.
    - Előnye, hogy egyszerre állítja be az új hely koordinátáit és az új méreteket, így a komponens csak egyszer rajzolja újra saját magát.

Kivételkezelés

- **Felhasználói programban szükséges a bemenetek ellenőrzése és a felmerülő hibalehetőségek kiküszöbölése**
- **Ennek általános eszköze az ún. kivételkezelés**
  - Bővebben PPT-n!
  - A program normális működésétől eltérő, váratlan, „kivételes” eseteket kezeljük vele



Kivételkezelés

```
try
{
    ...
}
catch
{
    ...
}
finally
{
    ...
}
```

- **A try blokkba helyezzük azt a részt, amely problémát okozhat**
- **A catch blokkba kerül a vezérlés, ha a try blokkban lévő kódban hiba történt**
- **A finally blokkban lévő rész mindenképp lefut**
  - A finally elhagyható

- **Honnan tudjuk egy kódrészről, hogy problémát okozhat?**
  - Józan ész
  - Felhasznált függvények ellenőrzése

```
float jovedelem = float.Parse(textBoxJovedelem.Text);
float ado = float.Parse(
float adoeloleg = float.

float befizetendo = jov

if (befizetendo < adoe
    labelHelyzet.Text =
else if (befizetendo >
```

float float.Parse(string s)
Converts the string representation of a number to its single-precision floating-point number equivalent.

Exceptions:
System.ArgumentNullException
System.FormatException
System.OverflowException

Kivételkezelés

```
try
{
    float jovedelem = float.Parse(textBoxJovedelem.Text);
}
catch (FormatException)
{
    labelHiba.Text = "Hibás formátum!";
}
catch (OverflowException)
{
    labelHiba.Text = "Túl nagy szám! ";
}
catch (Exception ex)
{
    labelHiba.Text = ex.Message;
}
```

- **Catch blokkból több is lehet, ezek közül csak egy fut le**
  - Megadjuk a hiba típusát, amit az adott catch blokk kezeljen
  - Az adott típusú vagy utód típusú kivételeket kapja el az adott blokk (PPT!)
  - Megadhatunk a típus mellett egy nevet is, ekkor a blokkon belül felhaszálhatjuk a kivétellel kapcsolatos információkat tartalmazó kivételobjektumot is

Hibakezelés más eszközei

- **Felhasználói bemenetek ellenőrzésére nagyon gyakran használjuk a TryParse()-t**
  - Parse() függvény hibakezeléssel ellátott változata

```
double jovedelem;
if (double.TryParse(textBoxJovedelem.Text, out jovedelem)
{
    ...
}
else labelHiba.Text = "Hibás bevitel!";
```

  - A visszatérési értékét kell ellenőrizni:
    - Ha igazat ad vissza, sikerült a konverzió, ekkor felhasználhatjuk az out paraméterként beadott értéket
    - Ha hamisat ad vissza, sikertelen volt a konverzió
      - Nem történik kivételdobás, a hiba okát nem tudjuk meg

- **A beviteli hibák megfelelő vezérlőválasztással és a bemenet szabályozásával részben kivédhetők**